

The image is a large, symmetrical, abstract graphic composed of the letters 'S' and 'Y' arranged in a grid-like pattern. The overall shape is a stylized 'Y' or a complex letter 'S'. The top part is a wide horizontal bar made of 'S's, with 'Y's forming a central vertical column. The sides are also made of 'S's, with 'Y's forming a central vertical column. The bottom part is a wide horizontal bar made of 'S's, with 'Y's forming a central vertical column. The entire graphic is composed of the letters 'S' and 'Y' arranged in a grid-like pattern.

```
BBBBBBBBB  UU  UU  GGGGGGGG  CCCCCCCC  HH  HH  EEEEEEEEE  CCCCCCCC  KK  KK
BBBBBBBBB  UU  UU  GGGGGGGG  CCCCCCCC  HH  HH  EEEEEEEEE  CCCCCCCC  KK  KK
BB  BB  UU  UU  GG  GG  CC  CC  HH  HH  EE  EE  CC  CC  KK  KK
BB  BB  UU  UU  GG  GG  CC  CC  HH  HH  EE  EE  CC  CC  KK  KK
BB  BB  UU  UU  GG  GG  CC  CC  HH  HH  EE  EE  CC  CC  KK  KK
BBBBBBBBB  UU  UU  GG  GG  CC  CC  HHHHHHHHHH  EEEEEEEEE  KKKKKK  KK
BBBBBBBBB  UU  UU  GG  GG  CC  CC  HHHHHHHHHH  EEEEEEEEE  KKKKKK  KK
BB  BB  UU  UU  GG  GG  CC  CC  HH  HH  EE  EE  CC  CC  KK  KK
BB  BB  UU  UU  GG  GG  CC  CC  HH  HH  EE  EE  CC  CC  KK  KK
BB  BB  UU  UU  GG  GG  CC  CC  HH  HH  EE  EE  CC  CC  KK  KK
BBBBBBBBB  UUUUUUUUU  GGGGGG  CCCCCCCC  HH  HH  EEEEEEEEE  CCCCCCCC  KK  KK
BBBBBBBBB  UUUUUUUUU  GGGGGG  CCCCCCCC  HH  HH  EEEEEEEEE  CCCCCCCC  KK  KK
KK  KK  ....
```

```
LL  I11111  SSSSSSSS
LL  I11111  SSSSSSSS
LL  II  SS
LL  II  SS
LL  II  SS
LL  II  SSSSSS
LL  II  SSSSSS
LL  II  SS
LL  II  SS
LL  II  SS
LL  I11111  SSSSSSSS
LL  I11111  SSSSSSSS
```

(1)	190	BUG CHECK ERROR MESSAGE PROCESSING
(1)	384	NON-RESIDENT BUG CHECK CODE
(1)	689	DUMP ARRAY - SUBROUTINE TO DUMP AN ARRAY OF MEMORY LOCATIONS
(1)	745	WRITEDUMP - WRITE DATA TO DUMP FILE
(1)	815	SUBROUTINES TO BUILD HEADERS AND VERIFY BOOT CONTROL BLOCK

```
0000 1 .TITLE BUGCHECK - SOFTWARE BUG CHECK ERROR LOGIC
0000 2 .IDENT 'V04-000'
0000 3
0000 4 *****
0000 5
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 * ALL RIGHTS RESERVED.
0000 9
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 * TRANSFERRED.
0000 16
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 * CORPORATION.
0000 20
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23
0000 24 *
0000 25 *****
0000 26
0000 27 D. N. CUTLER 29-OCT-77
0000 28
0000 29 SOFTWARE BUG CHECK ERROR LOGIC
0000 30
0000 31 MODIFICATION HISTORY:
0000 32
0000 33 V03-011 KTA3113 Kerbey T. Altmann 21-Mar-1984
0000 34 Add support for calling a SCS shutdown routine.
0000 35 Put in a halt if bugcheck code cannot be read.
0000 36 Add some more comments.
0000 37
0000 38 V03-010 MSH0008 Michael S. Harvey 10-Feb-1984
0000 39 Don't display image name if no image is active.
0000 40
0000 41 V03-009 KDM0049 Kathleen D. Morse 08-Jul-1983
0000 42 Move ICR, TODR, and ACCS to cpu-dependent register
0000 43 dump routine.
0000 44
0000 45 V03-008 KTA3060 Kerbey T. Altmann 22-Jun-1983
0000 46 Add code to call a possible unit disconnect routine
0000 47 in bootdriver after shutdown.
0000 48
0000 49 V03-007 ROW0188 Ralph O. Weber 30-APR-1983
0000 50 Fix truncation errors to ERL$ routines.
0000 51
0000 52 V03-006 TCM0003 Trudy C. Matthews 16-Feb-1983
0000 53 Initialize console registers in a CPU-dependent fashion
0000 54 before doing I/O to console terminal.
0000 55
0000 56 V03-005 TCM0002 Trudy C. Matthews 16-Dec-1982
0000 57 Initialize R2 before calling CON$SENDCONSCMD.
```

```
0000 58 :
0000 59 :
0000 60 : V03-004 TCM0001 Trudy C. Matthews 10-Nov-1982
0000 61 : Call CPU-dependent routine CON$SENDCONSCMD to send "reboot
0000 62 : CPU" command to the console.
0000 63 :
0000 64 : V03-003 ROW0120 Ralph O. Weber 24-AUG-1982
0000 65 : Change EXE$BOOTCB_CHK to not include the WCB$L_READS and the
0000 66 : WCB$L_WRITES fields of the SYS.EXE window control block in the
0000 67 : boot control block / SYS.EXE window control block checksum.
0000 68 : Paging I/O counts page reads/writes in these fields thus
0000 69 : causing a checksum test which includes them to fail
0000 70 : unnecessarily.
0000 71 :
0000 72 : V03-002 KDM0002 Kathleen D. Morse 28-Jun-1982
0000 73 : Added $IODEF.
0000 74 :
0000 75 : MACRO LIBRARY CALLS
0000 76 :
0000 77 :
0000 78 : $BOODEF ;DEFINE BOOT CONTROL BLOCK OFFSETS
0000 79 : $BQODEF ;DEFINE BOOT QIO OFFSETS
0000 80 : $CONDEF ;DEFINE CONSOLE FUNCTION CODES
0000 81 : $DMPDEF ;DEFINE DUMP FILE HEADER BLOCK
0000 82 : $EMBDEF <CR,BC> ;DEFINE EMB OFFSETS
0000 83 : $IFDDEF ;IMAGE FILE DESCRIPTOR DEFINITIONS
0000 84 : $IODEF ;DEFINE I/O FUNCTION CODES
0000 85 : $MBADEF ;MASS BUS ADAPTER INITIALIZATION
0000 86 : $PCBDEF ;DEFINE PCB OFFSETS
0000 87 : $PFNDEF ;DEFINE PFN DATA BASE BITS AND FIELDS
0000 88 : $PRDEF ;DEFINE PROCESSOR REGISTERS
0000 89 : $PRVDEF ;DEFINE PRIVILEGE BITS
0000 90 : $PTEDEF ;DEFINE PAGE TABLE BITS AND FIELDS
0000 91 : $PSLDEF ;DEFINE PROCESSOR STATUS BITS
0000 92 : $RPBDEF ;DEFINE RESTART PARAMETER BLOCK
0000 93 : $$SDEF ;DEFINE SYSTEM STATUS VALUES
0000 94 : $$STSDEF ;DEFINE STATUS CODE FIELDS
0000 95 : $UBADEF ;DEFINE UNIBUS ADAPTER VALUES
0000 96 : $VADEF ;DEFINE VIRTUAL ADDRESS FIELDS
0000 97 : $WCBDEF ;DEFINE WINDOW CONTROL BLOCK OFFSETS
0000 98 :
0000 99 :
0000 100 : LOCAL SYMBOLS
0000 101 :
00000000 102 : .PSECT $$$025
0000 103 :
00000000 104 BUGCHK_FLAGS: ;FLAGS TO BE USED BY BUGCHECK CODE
0000 105 .LONG 0
00000000 106 FATAL_SPSAV:
0000 107 .LONG 0 ;FATAL BUGCHECK IN PROGRESS SP
00000000 108 EXE$GL_BUGCHECK: ;SAVED FATAL BUGCHECK CODE
0000 109 .LONG 0
000C 110 :
000C 111 : CHARACTER CODE DEFINITIONS
000C 112 :
000C 113 :
0000000D 000C 114 CR=13 ;CARRIAGE RETURN
```

```
0000000A 000C 115 LF=10 ;LINE FEED
000C 116
000C 117
000C 118 : LOCAL DATA
000C 119
00000000 120 .PSECT $ZBUGFATAL,WORD ;PSECT TO LOCATE EXECUTION LOCATION FOR
0000 121 ;BUGCHECK
0000 122 BUG$FATAL:: ;MARKER ADDRESS
0000 123
00000000 124 .PSECT Z$INIT__BUGZEND,WORD ;END OF BUGCHECK PSECTS
0000 125 BUG$A_PAGEDEND::
0000 126
0000 127
0000 128 : BUG CHECK MESSAGE CONTROL TEXT
0000 129
0000 130
00000000 131 .PSECT Z$INIT__BUGC
0000 132 PRCNAM_MSG:
0000 133 .ASCII <CR><LF><LF>/ CURRENT PROCESS = /

52 52 55 43 20 20 20 20 0A 0A 0D 00' 0000
20 53 53 45 43 4F 52 50 20 54 4E 45 000C
20 3D 0018
19 0000
001A
43 4F 52 50 20 20 20 20 0A 0A 0D 00' 001A
47 45 4C 49 56 49 52 50 20 53 53 45 0026
0A 0A 0D 53 45 0032
1C 001A
0037
47 41 4D 49 20 20 20 20 0A 0A 0D 00' 0037
20 3D 20 45 4D 41 4E 20 45 0043
14 0037
004C
48 53 20 4D 45 54 53 59 53 09 0A 0D 004C
4C 50 4D 4F 43 20 4E 57 4F 44 54 55 0058
20 2D 20 45 54 45 0064
20 45 4C 4F 53 4E 4F 43 20 45 53 55 006A
54 53 59 53 20 54 4C 41 48 20 4F 54 0076
00 0A 0D 4D 45 0082
0087
41 54 41 46 20 2A 2A 2A 2A 0A 0A 0D 0087
2C 4B 43 45 48 43 20 47 55 42 20 4C 0093
00 20 3D 20 4E 4F 49 53 52 45 56 20 009F
00AB
45 54 53 49 47 45 52 20 20 20 20 0A 00AB
0A 0A 0D 50 4D 55 44 20 52 00B7
00 20 3D 20 30 52 09 00C0
00 20 3D 20 31 52 09 00C7
00 20 3D 20 32 52 09 00CE
00 20 3D 20 33 52 09 00D5
00 20 3D 20 34 52 09 00DC
00 20 3D 20 35 52 09 00E3
00 20 3D 20 36 52 09 00EA
00 20 3D 20 37 52 09 00F1
00 20 3D 20 38 52 09 00F8
00 20 3D 20 39 52 09 00FF
00 20 3D 30 31 52 09 0106
00 20 3D 31 31 52 09 010D

134 PRCPRV_MSG:
135 .ASCII <CR><LF><LF>/ PROCESS PRIVILEGES/<CR><LF><LF>

136 IMGNAM_MSG:
137 .ASCII <CR><LF><LF>/ IMAGE NAME = /

138 SHUT_DOWN:
139 .ASCII <CR><LF>/ ;OPERATOR REQUESTED SHUTDOWN
SYSTEM SHUTDOWN COMPLETE - / ;

140 .ASCIIZ /USE CONSOLE TO HALT SYSTEM/<CR><LF>

141 MSGCTRL:
142 .ASCIIZ <CR><LF><LF>/**** FATAL BUG CHECK, VERSION = / ;

143 MSGCTRL1:
144 .ASCII <LF>/ REGISTER DUMP/<CR><LF><LF> ;

145 .ASCIIZ / R0 = /
146 .ASCIIZ / R1 = /
147 .ASCIIZ / R2 = /
148 .ASCIIZ / R3 = /
149 .ASCIIZ / R4 = /
150 .ASCIIZ / R5 = /
151 .ASCIIZ / R6 = /
152 .ASCIIZ / R7 = /
153 .ASCIIZ / R8 = /
154 .ASCIIZ / R9 = /
155 .ASCIIZ / R10 = /
156 .ASCIIZ / R11 = /
```

```
00 20 3D 20 50 41 09 0114 157 .ASCII / AP = /  
00 20 3D 20 50 46 09 011B 158 .ASCII / FP = /  
00 20 3D 20 50 53 09 0122 159 .ASCII / SP = /  
00 20 3D 20 43 50 09 0129 160 .ASCII / PC = /  
00 20 3D 4C 53 50 09 0130 161 .ASCII / PSL = /  
2F 4C 45 4E 52 45 4B 20 20 20 20 0A 0137 162 .ASCII <LF>^ KERNEL/INTERRUPT STACK^<CR><LF><LF><128> ;  
54 53 20 54 50 55 52 52 45 54 4E 49 0143  
80 0A 0A 0D 4B 43 41 014F  
54 53 20 43 45 58 45 20 20 20 20 0A 0156 163 .ASCII <LF>/ EXEC STACK/<CR><LF><LF><128> ;  
80 0A 0A 0D 4B 43 41 0162  
0169 164  
0169 165  
0169 166  
0169 167  
0169 168  
00000000 169  
0000 170  
00 0000 171  
01 0001 172  
02 0002 173  
03 0003 174  
04 0004 175  
80 0005 176  
08 0006 177  
09 0007 178  
0A 0008 179  
0B 0009 180  
0C 000A 181  
0D 000B 182  
10 000C 183  
11 000D 184  
13 000E 185  
15 000F 186  
18 0010 187  
80 0011 188
```

PROCESSOR REGISTER DUMP CONTROL TABLE

REGTAB:	.PSECT	\$AEXENONPAGED
	.BYTE	PR\$_KSP
	.BYTE	PR\$_ESP
	.BYTE	PR\$_SSP
	.BYTE	PR\$_USP
	.BYTE	PR\$_ISP
	.BYTE	128
	.BYTE	PR\$_POBR
	.BYTE	PR\$_POLR
	.BYTE	PR\$_P1BR
	.BYTE	PR\$_P1LR
	.BYTE	PR\$_SBR
	.BYTE	PR\$_SLR
	.BYTE	PR\$_PCBB
	.BYTE	PR\$_SCBB
	.BYTE	PR\$_ASTLVL
	.BYTE	PR\$_SISR
	.BYTE	PR\$_ICCS
	.BYTE	128

:
: KERNEL STACK POINTER
: EXECUTIVE STACK POINTER
: SUPERVISOR STACK POINTER
: USER STACK POINTER
: INTERRUPT STACK POINTER
: TABLE ESCAPE
: PROGRAM REGION BASE REGISTER
: PROGRAM REGION LIMIT REGISTER
: CONTROL REGION BASE REGISTER
: CONTROL REGION LIMIT REGISTER
: SYSTEM BASE REGISTER
: SYSTEM LIMIT REGISTER
: PROCESS CONTROL BLOCK BASE REGISTER
: SYSTEM CONTROL BLOCK BASE REGISTER
: AST DELIVERY LEVEL REGISTER
: SOFTWARE INTERRUPT SUMMARY REGISTER
: INTERVAL TIMER CONTROL REGISTER
: TABLE ESCAPE

```
0012 190 .SBTTL BUG CHECK ERROR MESSAGE PROCESSING
0012 191 :+
0012 192 : EXE$BUG_CHECK - BUG CHECK ERROR MESSAGE PROCESSING
0012 193 :
0012 194 :
0012 195 : THIS ROUTINE IS CALLED BY EXECUTING THE OPERATION CODES ^XFEFF AND
0012 196 : ^FDFF, WHICH ARE RESERVED FOR DIGITAL AND ARE GUARANTEED TO ALWAYS
0012 197 : CAUSE AN EXCEPTION.
0012 198 :
0012 199 : THIS ROUTINE CONTAINS A HOOK FOR LOADABLE MULTI-PROCESSING CODE.
0012 200 : THE HOOK, MPH$BUGCHKHK, MUST BE LOCATED ON THE "JSB EXE$ADPINIT"
0012 201 : INSTRUCTION. AFTER EXECUTING SOME MULTI-PROCESSING SPECIFIC CODE,
0012 202 : EXECUTION WILL BE CONTINUED BY JUMPING TO EXE$ADPINIT AND THEN
0012 203 : RETURNING TO THE IN-LINE CODE IN THIS ROUTINE.
0012 204 :
0012 205 : INPUTS:
0012 206 :
0012 207 : THE CURRENT PROCESS PCB.
0012 208 : THE ENTIRE PROCESSOR STATE (I.E. GENERAL REGISTERS, ETC.).
0012 209 : THE BUG CHECK CODE WHICH FOLLOWS IMMEDIATELY INLINE.
0012 210 :
0012 211 : OUTPUTS:
0012 212 :
0012 213 : IF THE PREVIOUS MODE WAS KERNEL OR EXECUTIVE AND THE BUG SEVERITY IS
0012 214 : GREATER THAN OR EQUAL TO ERROR, THEN THE SYSTEM IS SHUT DOWN IN AN
0012 215 : ORDERLY FASHION BY EXECUTING THE CRASH RESTART ROUTINE. THE CODE
0012 216 : TO HANDLE A FATAL BUGCHECK IS READ FROM THE SYSTEM IMAGE OVER SOME
0012 217 : OF THE PURE EXEC CODE USING THE SAVED BOOTSTRAP DRIVER.
0012 218 :
0012 219 : IF THE PREVIOUS MODE WAS KERNEL OR EXECUTIVE AND THE BUG SEVERITY IS
0012 220 : LESS THAN ERROR, THEN AN ERROR LOG ENTRY IS MADE AND EXECUTION OF THE
0012 221 : SYSTEM CONTINUES.
0012 222 :
0012 223 : IF THE PREVIOUS MODE WAS SUPERVISOR OR USER AND THE PROCESS HAS THE
0012 224 : PRIVILEGE TO CAUSE BUG CHECK ERROR LOG ENTRIES, THEN AN ENTRY IS MADE
0012 225 : IN THE ERROR LOG. OTHERWISE NO ENTRY IS MADE.
0012 226 :
0012 227 : IF THE PREVIOUS MODE WAS SUPERVISOR OR USER AND THE BUG SEVERITY IS
0012 228 : GREATER THAN OR EQUAL TO ERROR, THEN AN EXIT SYSTEM SERVICE IS PERFORMED
0012 229 : ON BEHALF OF THE PROCESS AT THE MODE CAUSING THE BUG CHECK. IF THE BUG
0012 230 : SEVERITY IS LESS THAN ERROR, THEN EXECUTION OF THE PROCESS IS RESUMED.
0012 231 :
0012 232 : IF AN ACCESS VIOLATION IS DETECTED WHILE ATTEMPTING TO FETCH THE BUG
0012 233 : CHECK CODE, THE EXCEPTION IS TURNED INTO AN ACCESS VIOLATION.
0012 234 : -
0012 235 :
0012 236 EXE$BUG_CHECK:: ;BUG CHECK ERROR PROCESSING
0012 237 .ENABL LSB
0012 238 PUSHF #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP,SP> ;SAVE
0012 239 MOVL 15*4(SP),R0 ;GET ADDRESS OF INSTRUCTION
0012 240 IFRD #2,2(R0),20$ ;CAN LOWER HALF OF BUG CHECK CODE BE READ?
0012 241 10$: POPR #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP,SP> ;RESTORE
0012 242 PUSHF (SP) ;DUPLICATE ADDRESS OF INSTRUCTION
0012 243 ADDL #2,(SP) ;CALCULATE ADDRESS OF VIOLATION
0012 244 PUSHF #0 ;SET REASON CODE
0012 245 BRW EXE$ACVIOLAT
0012 246
```

7FFF 8F BB	0012 238
50 3C AE DO	0016 239
	001A 240
7FFF 8F BA	0021 241
6E DD	0025 242
6E 02 CO	0027 243
00 DD	002A 244
FFD1' 31	002C 245
	002F 246

```
002F 247 :  
002F 248 : BUG CHECK CODE CAN BE READ  
002F 249 :  
002F 250 :  
5D 02 A0 3C 002F 251 20$: MOVZWL 2(R0),FP ;GET LOWER HALF OF BUGCHECK CODE  
3C AE 04 C0 0033 252 ADDL #4,15*4(SP) ;CALCULATE ADDRESS OF NEXT INSTRUCTION  
5C 5E D0 0037 253 MOVL SP,AP ;SET ADDRESS OF SAVED REGISTERS  
5B C3 AF 9E 003A 254 MOVAB REGTAB,R11 ;GET ADDRESS OF PROCESSOR REGISTER TABLE  
01 A0 FD 8F DC 003E 255 MOVPSL R10 ;READ CURRENT PROCESSOR STATUS  
OF 12 0040 256 CMPB #XFD,1(R0) ;BUG CHECK LONG?  
0045 257 BNEQ 25$ ;IF NEQ NO  
0047 258 IFNORD #2,4(R0),10$ ;CAN UPPER HALF OF BUG CHECK CODE BE READ?  
5D 02 A0 D0 004E 259 MOVL 2(R0),FP ;GET BUG CHECK CODE  
3C AE 02 C0 0052 260 ADDL #2,15*4(SP) ;CALCULATE ADDRESS OF NEXT INSTRUCTION  
02 16 ED 0056 261 25$: CMPZV #PSL$V_PVMOD,#PSL$S_PVMOD,- ;PREVIOUS MODE EXEC OR KERNEL?  
01 5A 0059 262 R10,#PSL$C_EXEC  
64 15 005B 263 BLEQ 70$ ;IF LEQ YES  
54 0000'CF D0 005D 264 MOVL W^SCH$GL_CURPCB,R4 ;GET CURRENT PROCESS PCB ADDRESS  
0062 265 IFNPRIV BUGCHK,40$ ;DOES PROCESS HAVE PRIVILEGE TO BUG CHECK?  
59 0070 8F 3C 0068 266 MOVZWL #EMBSK_UBC,R9 ;SET ENTRY TYPE  
51 0080 8F 3C 006D 267 30$: MOVZWL #EMBSK_BC_LENGTH,R1 ;GET LENGTH OF BUGCHECK MESSAGE  
00000000'EF 16 0072 268 JSB ERL$ALLOCEMB ;ALLOCATE BUG CHECK ERROR MESSAGE BUFFER  
23 50 E9 0078 269 BLBC R0,40$ ;IF LBC ALLOCATION FAILURE  
00FA 30 007B 270 BSBW BUILD HEADER ;BUILD MESSAGE HEADER AND DUMP REGISTERS  
68 A2 5D D0 007E 271 MOVL FP,EMBSL_BC_CODE(R2) ;SET BUGCHECK CODE INTO MESSAGE  
51 00000000'9F D0 0082 272 MOVL @#SCH$GL_CURPCB,R1 ;GET ADR OF CURRENT PROCESS'S PCB  
60 A1 D0 0089 273 MOVL PCB$S_PID(R1),- ;SET PROCESS ID INTO MESSAGE  
6C A2 008C 274 EMBSL_BC_PID(R2)  
70 A1 7D 008E 275 MOVQ PCB$S_LNAME(R1),- ;SET PROCESS NAME INTO  
70 A2 0091 276 EMBSL_BC_LNAME(R2)  
78 A1 7D 0093 277 MOVQ PCB$S_LNAME+8(R1),- ;SET PROCESS NAME INTO  
78 A2 0096 278 EMBSL_BC_LNAME+8(R2) ; ERROR LOG MESSAGE  
00000000'EF 16 0098 279 JSB ERL$RELEASEMB ;RELEASE ERROR MESSAGE BUFFER  
0C 5D E8 009E 280 40$: BLBS FP,50$ ;IF LBS NONFATAL BUG CHECK  
03 00 ED 00A1 281 CMPZV #ST$S$V_SEVERITY,#ST$S$S_SEVERITY,- ;FATAL BUG CHECK?  
02 5D 00A4 282 FP,#ST$S$K_ERROR  
05 19 00A6 283 BLSS 50$ ;IF LSS NO  
3C AE B2'AF 9E 00A8 284 MOVAB B^60$,15*4(SP) ;REPLACE RETURN PC  
7FFF 8F BA 00AD 285 50$: POPR #M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP,SP> ;RESTORE  
02 00B1 286 REI  
00B2 287 :  
00B2 288 :  
00B2 289 : EXECUTE EXIT SYSTEM SERVICE ON BEHALF OF PROCESS  
00B2 290 :  
00B2 291 :  
F1 11 00B2 292 60$: $EXIT_S #SS$S_BUGCHECK ;EXIT MODE  
00BF 293 BRB 60$  
00C1 294 :  
00C1 295 :  
00C1 296 : PREVIOUS MODE WAS EXECUTIVE OR KERNEL  
00C1 297 :  
00C1 298 :  
0A 0000'CF 59 28 3C 00C1 299 70$: MOVZWL #EMBSK_SBC,R9 ;SET ENTRY TYPE  
00 00 E0 00C4 300 BBS S^#EXE$V_FATAL_BUG,W^EXE$GL_FLAGS,75$ ;IF SET, ALL FATAL  
A0 5D E8 00CA 301 BLBS FP,30$ ;IF LBS NONFATAL BUG CHECK  
03 00 ED 00CD 302 CMPZV #ST$S$V_SEVERITY,#ST$S$S_SEVERITY,- ;FATAL BUG CHECK?  
02 5D 00D0 303 FP,#ST$S$K_ERROR ;
```

```

          99  19 00D2 304      BLSS 30$      ;IF LSS NO
          00C0 30 00D4 305
          03 13 00D4 306 75$: BSBW EXE$BOOTCB_CHK ;IS BOOT CONTROL BLOCK OK?
          24 A1 D4 00D7 307      BEQL 80$      ;BRANCH IF YES
          00DC 308      CLRL BOO$L_BUG_MAP(R1) ;NO, SET UP TO ISSUE REBOOT
          00DC 309
          00DC 310 : SHUT DOWN SYSTEM IN AN ORDERLY MANNER
          00DC 311 :
          00DC 312 :
          00DC 313 :
          06 0000'CF 00 E2 00DC 314 80$: SETIPL #31 ;DISABLE ALL INTERRUPTS
          00000000'GF 16 00DF 315      BBSS #0,W^BUGCHK_FLAGS,82$ ;ONLY DO THIS ONCE
          56 0000'CF D0 00E5 316      JSB G^SCS$SHUTDOWN ;CALL SCS DO SHUTDOWN ALL CIRCUITS
          00000000'GF 16 00EB 317 82$: MOVL W^EXE$GL_RPB,R6 ;GET ADDRESS OF RESTART PARAMETERS
          00000000'GF 16 00F0 318      JSB G^EXE$SHOTDWNADP ;SHUT DOWN ANY ADAPTERS THAT NEED IT
          00000000'GF 16 00F6 319 MPH$BUGCHKHK:: ;MULTI-PROCESSING HOOK (REPLACES JSB)
          00FC 320      JSB G^EXE$INIBOOTADP ;INIT BOOT DEVICE ADAPTER BEFORE
          00FC 321      ; READING FATAL BUGCHECK CODE
          00FC 322      ; DO NOT STEP OVER NEXT 2 LINES OR PAGES
          00FC 323      ; WILL BE SET RONLY AFTER JSB BY XDELTA
          00 00000000'EF 16 00FC 324      JSB INIS$WRITABLE ;MAKE SYSTEM CODE WRITABLE
          00 0000'CF 00' E2 0102 325      BBSS S^#EXE$V_SYSWRTABL,W^EXE$GL_FLAGS,85$ ; INHIBIT INIS$RONLY/WRITABLE
          53 34 A6 D0 0108 326      ; (RONLY CALLED ON EVERY XDELTA EXIT)
          010C 327 85$: MOVL RPB$L_IOVEC(R6),R3 ;FETCH POINTER TO BOOTDRIVER
          010C 328 :
          010C 329 : CHECK THE VMB VERSION NUMBER. IF IT EXISTS AND IF IT IS 7 OR GREATER, THEN
          010C 330 : CALL A UNIT INITIALIZATION ROUTINE TO DO ANY DEVICE/UNIT SPECIFIC INIT
          010C 331 : THAT IS NOT DONE IN ADAPTER INIT.
          010C 332 :
          50 10 A3 B2 010C 333      MCOMW BQO$W_VERSION(R3),R0 ;GET VMB VERSION NUMBER 1'S COMPLEMENTED
          12 A3 50 B1 0110 334      CMPW R0,BQO$W_VERSION+2(R3) ;CHECK AGAINST CHECK WORD IN VMB
          07 10 A3 B1 0114 335      BNEQ 90$ ;IF NOT, ASSUME NO VERSION NUMBER
          55 1C A3 D0 0116 336      CMPW BQO$W_VERSION(R3),#7 ;VERSION 7 OR GREATER OF VMB?
          59 0A 13 011A 337      BLSSU 90$ ;NO, DON'T CALL THE NON-EXISTENT CODE
          6345 00 FB 0120 338      MOVL BQO$L_UNIT_INIT(R3),R5 ;YES, IS THE ROUTINE PRESENT?
          32 50 E9 0122 339      BEQL 90$ ;NO, DON'T CALL
          55 00000000'GF D0 0125 340      MOVL R6,R9 ;YES, SHIFT INPUT PARAMETERS
          52 24 A5 D0 0129 341      CALLS #0,(R3)[R5] ;DO IT!
          25 13 0137 342      BLBC R0,REBOOT ;INIT FAILED, JUST REBOOT
          56 DD 0139 343 90$: MOVL C^EXE$GL_BOOTCB,R5 ;ADDRESS OF BOOT CONTROL BLOCK
          01 DD 013B 344      MOVL BOO$L_BUG_MAP(R5),R2 ;VIRTUAL TO LOGICAL MAP FOR
          21 DD 013D 345      ;NON-RESIDENT BUGCHECK CODE AND DATA
          08 A2 DD 013F 346      BEQL REBOOT ;REBOOT IF BAD BOOT CONTROL BLOCK
          7E 04 A2 09 78 0142 347      PUSHL R6 ;SET RPB ADDRESS IN ARGUMENT LIST
          0000'CF 9F 0147 348      PUSHL #1 ;SET FOR VIRTUAL ADDRESS I/O
          00 B343 6E FA 014B 349      PUSHL S^#IOS_READLBLK ;SET FUNCTION TO READ
          03 50 E9 0152 350      PUSHL 8(R2) ;STARTING LBN
          04 BE 17 0155 351      ASHL #9,4(R2),-(SP) ;NO. OF BYTES IN THIS PIECE
          5E 1C C0 0158 352      PUSHAB W^BUG$FATAL ;BUFFER ADDRESS
          FF7E 31 015B 353      PUSHL #6 ;NO. OF ARGUMENTS
          00 0158 354      CALLG (SP),@BQO$L_QIO(R3)[R3] ;CALL BOOTDRIVR TO READ FOLLOWING CODE
          03 50 E9 0152 355      BLBC R0,READ_ERR_RETRY ;BR IF ERROR TO RETRY
          04 BE 17 0155 356      JMP @4(SP) ;JUMP TO FATALBUG
          5E 1C C0 0158 357      READ_ERR_RETRY:
          FF7E 31 015B 358      ADDL #7*4,SP ;CLEAN OFF ARG LIST
          00 0158 359      BRW 80$ ;TRY READ AGAIN
          00 015B 360
```

```
015E 361
015E 362      .DSABL LSB
015E 363
015E 364      REBOOT THE PROCESSOR
015E 365
015E 366      Control comes here on 4 paths:
015E 367      1) Boot control block failed its checksum-verification
015E 368      2) Failure to initialize boot device/path
015E 369      3) Failure attempting to read first block of bugcheck code
015E 370      4) Dump terminated successfully and BUGREBOOT was on
015E 371
015E 372 REBOOT:
015E 373      BBC      S^#EXESV BUGREBOOT,-
0160 374      @#EXESGC FLAGS,10$      ;BRANCH IF NO REBOOT
0166 375      MOVL   #CONSC_BOOTCPU,R0      ;CONSOLE FUNCTION = REBOOT
0169 376      CLRL   R2      ;SIGNAL NO RETURN DATA EXPECTED
016B 377      JSB    CON$SENDCONSCMD      ;CALL CPU-DEPENDENT ROUTINE
0171 378
0171 379      CONTROL NEVER RETURNS HERE.
0171 380
00000000'9F 16 0171 381 10$: JSB @#INISBRK      ;STOP IN XDELTA IF PRESENT
00 0177 382      HALT
```

```
0178 384 .SBTTL NON-RESIDENT BUG CHECK CODE
0178 385
0178 386
0178 387 : READ IN THE REST OF THE BUGCHECK CODE AND DATA THAT WAS NOT CONTIGUOUS
0178 388 : WITH THIS FIRST PART. THE FOLLOWING CODE MUST BE TOTALLY CONTAINED
0178 389 : IN THE FIRST PAGE OF THE NON-RESIDENT BUGCHECK CODE TO BE CERTAIN
0178 390 : THAT IT IS READ BY THE FIRST READ IN THE RESIDENT PORTION.
0178 391
0178 392 : THE FOLLOWING STATE IS ASSUMED:
0178 393 : R2 = VIRTUAL TO LOGICAL MAP FOR NON-RESIDENT BUGCHECK CODE AND DATA
0178 394 : R3 = RPB$L_IOVEC(RPB)
0178 395 : THE FIRST SEVEN LONG WORDS ON THE STACK ARE THE ARGUMENT LIST
0178 396 : TO BOO$QIO IN THE BOOT DRIVER.
0178 397
0178 398
00000000 399 .PSECT Z$INIT,_BUGA,PAGE ;FIRST BUGCHECK PSECT IN INIT REGION
0000 400
0000 401 BUG$A PAGED::
0000 402 FATALBUG: ;START OF FATAL BUGCHECK CODE
57 82 FD 8F 78 0000 403 ASHL #-3,(R2)+,R7 ;GET COUNT OF RETRIEVAL POINTERS
52 08 C0 0005 404 ADDL #8,R2 ;POINT TO SECOND RETRIEVAL POINTER
1C 11 0008 405 BRB 30$ ;ALREADY DONE FIRST POINTER
04 AE 08 AE C0 000A 406 20$: ADDL 8(SP),4(SP) ;ADJUST XFER ADR BY BYTE COUNT READ
08 AE B2 09 78 000F 407 ASHL #9,(R2)+,8(SP) ;BYTE COUNT FOR NEXT PIECE
0C AE 82 D0 0014 408 MOVL (R2)+,12(SP) ;LBN FOR NEXT PIECE
00 B343 6E FA 0018 409 CALLG (SP),@BOO$QIO(R3)[R3] ;READ BUGCHECK CODE AND DATA
06 50 E8 001D 410 BLBS R0,30$ ;BRANCH IF OK
00000158'9F 17 0020 411 JMP @#READ_ERR_RETRY ;ERROR - TRY THE WHOLE THING OVER
E1 57 F5 0026 412 30$: SOBGTR R7,20$ ;READ EVERYTHING IN THE MAP
5E 1C C0 0029 413 ADDL #7*4,SP ;CLEAN OFF THE ARG LIST
002C 414
002C 415 : END OF CODE THAT MUST BE TOTALLY CONTAINED IN THE FIRST PAGE OF
002C 416 : NON-RESIDENT BUGCHECK CODE.
002C 417
50 00000004'9F 9E 002C 418 MOVAB @#FATAL_SPSAV,R0 ;ADDRESS OF SAVED FATAL SP
60 D5 0033 419 TSTL (R0) ;ALREADY IN A FATAL BUGCHECK?
06 13 0035 420 BEQL 82$ ;BRANCH IF NOT
5E 60 D0 0037 421 MOVL (R0),SP ;RESTORE SP FROM PREVIOUS BUGCHECK
01AA 31 003A 422 BRW CONSOLE_DONE ;AND GO REBOOT THE SYSTEM
60 5E D0 003D 423 82$: MOVL SP,(R0) ;NOTE THAT WE ARE IN A FATAL BUGCHECK
00 5A 16 E2 0040 424 BBSS #PSL$V_PVMOD,R10,84$ ;JAM PREVIOUS MODE TO EXEC
0044 425 ;THUS FORCING EXEC STACK DUMP TOO
0044 426
0044 427 : NOW BUILD THE DUMP FILE HEADER BLOCK. A PIECE OF SYSTEM SPACE IS
0044 428 : USED FOR THE BUFFER SINCE THIS IS THE ONLY ADDRESSES FOR WHICH I/O
0044 429 : CAN BE DONE. THE CRASH ERROR LOG ENTRY IS BUILT IN THIS BUFFER TO
0044 430 : GUARANTEE THAT IS IS INCLUDED IN THE DUMP, (SINCE THE ERROR LOG BUFFERS
0044 431 : MAY BE FULL).
0044 432
0044 433 84$: MOVAB FATALBUG-512+DMP$C_LENGTH+EMB$K_LENGTH,- ;BUFFER ADDRESS IS
0048 434 R2 ;THE PAGE PREVIOUS TO THIS CODE
0049 435 MOVL FP,EMB$C_CR_CODE(R2) ;SET BUGCHECK CODE INTO MESSAGE
004E 436 MOVL @#SCH$GL_CURPCB,R1 ;GET ADR OF CURRENT PROCESS'S PCB
0055 437 MOVL PCB$C_PID(R1),EMB$C_CR_PID(R2) ;SET PROCESS ID INTO MESSAGE
0058 438 MOVQ PCB$C_LNAME(R1),EMB$C_CR_LNAME(R2) ;SET PROCESS NAME INTO
0061 439 MOVQ PCB$C_LNAME+8(R1),EMB$C_CR_LNAME+8(R2) ;ERROR LOG MESSAGE
FC A2 010C 8F 3C 0067 440 MOVZWL #EMB$C_CR_LENGTH,EMB$C_SIZE(R2) ;SET THE SIZE OF THIS MSG
```

```
06 A2 00000000'9F 62 3E DB 006D 441 MFPR #PR$ SID,EMBSL HD SID(R2) ;SET SYSTEM ID IN MESSAGE
OE A2 00000000'9F 7D 0070 442 MOVQ @#EXESGQ_SYSTIME,EMBSQ CR TIME(R2) ;SET TIME ERROR OCCURRED
00000000'9F B0 0078 443 MOVW @#ERL$GL_SEQUENCE,EMBSQ CR ERRSEQ(R2) ;SET ERROR SEQUENCE NUMBER
00000000'9F D6 0080 444 INCL @#ERL$GL_SEQUENCE ;INCREMENT ERROR SEQUENCE NUMBER
59 25 3C 0086 445 MOVZWL #EMBSK CR R9 ;SET TYPE OF ENTRY
00000178'9F 16 0089 446 JSB @#BUILD_HEADER ;BUILD HEADER AND DUMP REGISTERS
0000018C'9F 16 008F 447 JSB @#DUMP_REGISTERS ;DUMP REMAINDER OF CPU-INDEPENDENT
00000000'9F 16 0095 448 ;PROCESSOR REGISTERS
00000000'9F 16 0095 449 JSB @#EXESDUMPCPUREG ;DUMP CPU-DEPENDENT PROCESSOR
00000000'9F FF A2 96 009B 450 ;REGISTERS
00000008'9F 5D D0 009E 451 INCB EMBSB VALID(R2) ;INDICATE ERL ENTRY IS COMPLETE
00000000'9F 16 00A5 452 MOVL FP,@#EXESGL_BUGCHECK ;SAVE BUGCHECK CODE
00000004'8F 5D D1 00AB 453 JSB @#CONSOWNCTY ;SET UP CONSOLE TERMINAL REGISTERS
03 12 00B2 454 CMPL FP,#<BUG$_OPERATOR!ST$SEVERE> ;IS THIS AN OPERATOR SHUTDOWN?
0130 31 00B4 455 BNEQ 100$ ;NO, CONTINUE
00B7 456 BRW 100$ ;YES, DONT GIVE NORMAL BUGCHECK MESSAGE
00B7 457 ;
00B7 458 ; OUTPUT THE BUGCHECK MESSAGE, REGISTER, AND STACK DUMP ON CONSOLE.
00B7 459 ;
59 5D DD 00B7 460 100$: PUSHL FP ;SAVE BUG CHECK CODE
5C 5E D0 00B9 461 MOVL SP,AP ;SET ADDRESS OF REGISTERS
5B D4 00BC 462 CLRL R11 ;SET FOR CONSOLE TERMINAL OUTPUT
00000087'EF 9E 00BE 463 MOVAB MSGCTRL,R9 ;GET ADDRESS OF CONTROL TEXT
50 89 98 00C5 464 110$: CVTBL (R9)+,R0 ;GET NEXT BYTE FROM CONTROL TEXT
6B 19 00C8 465 BLSS 130$ ;IF LSS END OF TEXT
05 13 00CA 466 BEQL 120$ ;IF EQL ESCAPE CHARACTER
FF31' 30 00CC 467 BSBW EXESOUTCHAR ;OUTPUT CHARACTER
F4 11 00CF 468 BRB 110$ ;
51 8C D0 00D1 469 120$: MOVL (AP)+,R1 ;GET NEXT LONGWORD TO CONVERT
50 00AB'CF 9E 00D4 470 MOVAB W^MSGCTRL1,R0 ;GET ADDRESS OF REGISTER STRING
59 50 D1 00D9 471 CMPL R0,R9 ;CHECK FOR END OF HEADER
47 12 00DC 472 BNEQ 124$ ;BRANCH IF NOT AT END
51 7E 7E 00DE 473 MOVAQ -(SP),R1 ;CREATE BUFFER FOR VERSION TEXT
01 AE 00000000'9F D0 00E1 474 MOVL @#SYS$GQ_VERSION,1(SP) ;SET VERSION NUMBER IN BUFFER
6E 05 90 00E9 475 MOVW #5,(SP) ;SET COUNT FOR VERSION AND SPACE
05 AE 20 90 00EC 476 MOVW #32,5(SP) ;SET TRAILING SPACE
FF0D' 30 00F0 477 BSBW EXESOUTCSTRING ;PRINT VERSION NUMBER
5E 08 C0 00F3 478 ADDL #8,SP ;CLEAN STACK
50 5D 08 C7 00F6 479 DIVL3 #8,FP,R0 ;CONVERT CODE TO INDEX
51 00000000'EF 9E 00FA 480 MOVAB BUG$T_MESSAGES,R1 ;SET BASE OF MESSAGES
52 81 9A 0101 481 122$: MOVZBL (R1)+,R2 ;GET LENGTH OF MESSAGE
51 52 C0 0104 482 ADDL R2,R1 ;AND POINT TO NEXT MESSAGE
F7 50 F5 0107 483 SOBGR R0,122$ ;BRANCH IF MESSAGE NOT LOCATED
FEF3' 30 010A 484 BSBW EXESOUTCSTRING ;OUTPUT STRING
51 00000000'EF DE 010D 485 MOVAL PRCNAM MSG,R1 ;"CURRENT PROCESS = "
FEE9' 30 0114 486 BSBW EXESOUTCSTRING ;OUTPUT COUNTED STRING
51 00000000'9F D0 0117 487 MOVL @#SCH$GL_CURPCB,R1 ;PROCESS PCB OF CURRENT PROCESS
51 00000070'8F C0 011E 488 ADDL #PCBST_LNAME,R1 ;POINT AT PROCESS NAME
FED8' 30 0125 489 BSBW EXESOUTCSTRING ;OUTPUT PROCESS NAME COUNTED STRING
FED5' 30 0128 490 BSBW EXESOUTRLF ;NEW LINE
98 11 012B 491 BRB 110$ ;
FED0' 30 012D 492 124$: BSBW EXESOUTHEX ;OUTPUT CONVERTED HEX LONGWORD
FED0' 30 0130 493 BSBW EXESOUTRLF ;OUTPUT CARRIAGE RETURN, LINE FEED PAIR
90 11 0133 494 BRB 110$ ;
58 40 8F 9A 0135 495 126$: MOVZBL #64,R8 ;SET LOOP COUNT
50 00000000'9F 9E 0139 496 130$: MOVAB @#C$LSAL_STACK,R0 ;POINTER TO POSSIBLE PROCESS SPACE STACKS
497
```

```
51 00000000'9F 9E 0140 498 MOVAB @#CTLSAL_STACKLIM,R1 ;POINTER TO POSSIBLE PROCESS STACK LIMIT
    12 5C 1F E1 0147 499 BBC #31,AP,135$ ;BRANCH IF STACK IS IN PROCESS SPACE
50 00000000'9F 9E 014B 500 MOVAB @#EXESAL_STACKS,R0 ;POINTER TO POSSIBLE SYSTEM SPACE STACKS
    51 FC A0 DE 0152 501 MOVAL -4(R0),RT ;USE SAME ARRAY AS LIMIT
    80 5C D1 0156 502 CMPL AP,(R0)+ ;ADDRESS IN FIRST(NULL) STACK?
    17 1B 0159 503 BLEQU 140$ ;YES, OKAY
    OA 11 015B 504 BRB 137$ ;NO, CHECK FURTHER
    015D 505
    015D 506 :: CHECK PROCESS KERNEL/EXEC STACKS
    80 5C D1 015D 507 135$: CMPL AP,(R0)+ ;ADDRESS IN FIRST STACK?
    05 1A 0160 508 137$ BGTRU 137$ ;NO, TOO HIGH - TRY SECOND(EXEC)
    61 5C D1 0162 509 CMPL AP,(R1) ;BELOW FIRST STACK LIMIT?
    0B 1A 0165 510 140$ BGTRU 140$ ;NO, ALL OKAY
    80 5C D1 0167 511 137$: CMPL AP,(R0)+ ;IN SECOND STACK?
    1A 1A 016A 512 155$ BGTRU 155$ ;BRANCH IF NOT, BAD STACK ADDRESS
    04 A1 5C D1 016C 513 CMPL AP,4(R1) ;NOW CHECK LIMIT
    14 1B 0170 514 155$ BLEQU 155$ ;NO, BAD STACK
50 70 5C C3 0172 515 140$: SUBL3 AP,-(R0),R0 ;NUMBER OF BYTES TO TOP OF STACK
    0E 15 0176 516 155$ BLEQ 155$ ;BRANCH IF EMPTY
    50 04 C6 0178 517 DIVL #4,R0 ;FORM LONG WORD COUNT OF MAX TO DUMP
    58 50 D1 017B 518 CMPL R0,R8 ;USE SMALLER OF MAX AND DEFAULT
    03 1B 017E 519 145$ BGEQ 145$
    58 50 D0 0180 520 MOVL R0,R8 ;USE THE MAX
    0182 30 0183 521 145$: BSBW DUMP_ARRAY ;DUMP KERNEL, INTERRUPT, OR EXEC STACK
    5B 5C 1F E0 0186 522 155$: BBS #31,AP,190$ ;DO NOT TRY FOR EXEC STACK IF SYSTEM SPACE
    5C 01 DB 018A 523 MFR #PR$ ESP,AP ;FETCH EXEC STACK POINTER
    A2 5A 16 E4 018D 524 BBSC #PSL$V_PVMOD,R10,126$ ;IF HAVEN'T DUMPED EXEC STACK, DO IT NOW
51 0000001A'EF 9E 0191 525 MOVAB PRCPRV_MSG,R1 ;"PROCESS PRIVILEGES"
    FE65' 30 0198 526 BSBW EXESOUTCSTRING ;OUTPUT COUNTED STRING
5C 00000000'9F D0 019B 527 MOVL @#SCH$GL_CURPCB,AP ;CURRENT PROCESS CONTROL BLOCK ADDRESS
    5C 6C AC D0 01A2 528 MOVL PCBSL_PHD(AP),AP ;PROCESS HEADER ADDRESS
    06 1B 01A6 529 BGEQ 170$ ;IF NOT NEGATIVE, DON'T TRY TO USE IT
    58 02 D0 01A8 530 MOVL #2,R8 ;2 LONG WORDS AT FRONT OF HEADER
    015A 30 01AB 531 BSBW DUMP_ARRAY ;OUTPUT THE PROCESS PRIVILEGES
51 00000037'EF 9E 01AE 532 170$: MOVAB IMGNAM_MSG,R1 ;"IMAGE NAME = "
    FE48' 30 01B5 533 BSBW EXESOUTCSTRING ;OUTPUT THE COUNTED STRING
5C 00000000'9F 9E 01B8 534 MOVAB @#CTLSGL_IMGHDRBF,AP ;GET POINTER TO IMAGE HEADER BUFFER
    58 D4 01BF 535 CLRL R8 ;DO NOT DUMP ANY DATA
    0144 30 01C1 536 BSBW DUMP_ARRAY ;JUST CHECK FOR ACCESSABILITY
    1B 50 E9 01C4 537 BLBC R0,180$ ;BRANCH IF CANNOT ACCESS THE POINTER
    5C 6C D0 01C7 538 MOVL (AP),AP ;GET IMAGE HEADER BUFFER ADDRESS
    16 13 01CA 539 BEQL 180$ ;IF EQL, NO IMAGE CURRENTLY ACTIVE
    58 D4 01CC 540 CLRL R8 ;DO NOT DUMP ANY DATA
    0137 30 01CE 541 BSBW DUMP_ARRAY ;JUST CHECK FOR ACCESSABILITY
    0E 50 E9 01D1 542 BLBC R0,180$ ;BRANCH IF CANNOT ACCESS THE IMAGE HDR BUF
51 04 AC D0 01D4 543 MOVL 4(AP),R1 ;ADDRESS OF IMAGE FILE DESCRIPTOR
50 02 A1 3C 01D8 544 MOVZWL IFD$W_FILNAMOFF(R1),R0 ;OFFSET TO NAME OF IMAGE BEING RUN
    51 50 C0 01DC 545 ADDL R0,R1 ;ADDRESS OF ASCII NAME
    FE1E' 30 01DF 546 BSBW EXESOUTCSTRING ;OUTPUT THE IMAGE NAME
    FE1B' 30 01E2 547 BSBW EXESOUTCRLF ;OUTPUT CARRIAGE RETURN, LINE FEED PAIR
    8E D5 01E5 548 180$: BSBW (SP)+ ;REMOVE BUG CHECK CODE FROM STACK
    01E7 549 190$: TSTL
    01E7 550
    01E7 551 :: OUTPUT TO CONSOLE, IF ANY, IS FINISHED. NOW WRITE OUT THE DUMP FILE.
    01E7 552
    01E7 553
    7FFF 8F BA 01E7 554 CONSOLE_DONE: POPR #*M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP,SP> ;RESTORE
```

```
06 00000000'00' E0 01EB 555 BBS S^#EXESV BUGREBOOT - ;CHECK FOR REBOOT
    00000000'9F 01ED 556      S^#EXESGL_FLAGS,10$
    00000000'9F 16 01F3 557 JSB 0^#INISBRK ;STOP IN XDELTA IF PRESENT
    7FFF 8F BB 01F9 558 10$:      ;CHECK AND WRITE THE DUMP FILE
    01FD 560      ;R7,R8,R9,R10,R11,AP,FP,SP>;STACK REGS
    01FD 561      ;FOR DISK WRITE
    01FF 562      ; BRANCH IF NO DUMP
    0205 563      ;
    0205 564      ; THE BOOT CONTROL BLOCK HAS ALREADY BEEN VALIDATED, JUST CHECK THAT
    0205 565      ; A DUMP FILE IS ACTUALLY PRESENT.
    0205 566      ;
5A 00000000'9F D0 0205 567      MOVL 0^#EXESGL BOOTCB,R10 ;BOOT CONTROL BLOCK ADDRESS
    55 20 AA D0 020C 568      MOVL BOO$SL_DMP_MAP(R10),R5 ;VIRTUAL TO LOGICAL MAP FOR DUMP FILE
    59 1C AA D0 0210 569      MOVL BOO$SL_DMP_SIZE(R10),R9 ;SIZE OF DUMP FILE IN BLOCKS
    03 14 0214 570      BGTR 30$ ;BRANCH IF SOME BLOCKS ARE PRESENT
    00AE 31 0216 571 20$:      BRW NODUMP ;NO DUMP
56 00000000'9F D0 0219 572 30$:      MOVL 0^#EXESGL_RPB,R6 ;GET BASE OF RESTART PARAMETER BLOCK
    55 34 A6 D0 0220 573      MOVL RPB$SL_IOVEC(R6),R5 ;FETCH POINTER TO BOOTDRIVER
    53 FBD8 CF 9E 0224 574      MOVAB FATALBUG-512,R3 ;GET ADDRESS OF DUMP HEADER BLK BUFFER
    57 53 D0 0229 575      MOVL R3,R7 ;SET BUFFER ADDRESS FOR WRITEDUMP
    022C 576      ASSUME DMP$SL_ERRSEQ EQ 0
83 00000000'9F D0 022C 577      MOVL 0^#ERL$GL_SEQUENCE,(R3)+ ;SAVE ERROR LOG SEQUENCE NUMBER
    83 B4 0233 578      ASSUME DMP$W_FLAGS EQ DMP$SL_ERRSEQ+4
    83 02 B0 0233 579      CLRW (R3)+ ;SET DUMP FILE FLAGS
    83 0C DB 0235 580      ASSUME DMP$W_DUMPVER EQ DMP$W_FLAGS+2
    83 0D DB 0235 581      MOVW #2,(R3)+ ;SET DUMP FILE VERSION NUMBER
    83 00 DB 0238 582      ASSUME DMP$SL_SBR EQ DMP$W_DUMPVER+2
    83 01 DB 0238 583      MFPR #PR$SBR,(R3)+ ;SET SYSTEM BASE REGISTER
    83 02 DB 023B 584      ASSUME DMP$SL_SLR EQ DMP$SL_SBR+4
    83 03 DB 023B 585      MFPR #PR$SLR,(R3)+ ;SET SYSTEM LENGTH REGISTER
    83 04 DB 023E 586      ASSUME DMP$SL_KSP EQ DMP$SL_SLR+4
    83 05 DB 023E 587      MFPR #PR$KSP,(R3)+ ;SET KERNEL STACK POINTER
    83 06 DB 0241 588      ASSUME DMP$SL_ESP EQ DMP$SL_KSP+4
    83 07 DB 0241 589      MFPR #PR$ESP,(R3)+ ;SET EXEC STACK POINTER
    83 08 DB 0244 590      ASSUME DMP$SL_SSP EQ DMP$SL_ESP+4
    83 09 DB 0244 591      MFPR #PR$SSP,(R3)+ ;SET SUPER STACK POINTER
    83 0A DB 0247 592      ASSUME DMP$SL_USP EQ DMP$SL_SSP+4
    83 0B DB 0247 593      MFPR #PR$USP,(R3)+ ;SET USER STACK POINTER
    83 0C DB 024A 594      ASSUME DMP$SL_ISP EQ DMP$SL_USP+4
    83 0D DB 024A 595      MFPR #PR$ISP,(R3)+ ;SET INTERRUPT STACK POINTER
    024D 596      ;
    024D 597      ; IF THE RPB WAS CREATED BY A VERSION OF VMB LESS THAN 3, THEN
    024D 598      ; CREATE A DUMMY MEMORY DESCRIPTOR FOR MAIN MEMORY BY ASSUMING
    024D 599      ; THAT THE SPT RESIDES AT THE END OF PHYSICAL MEMORY
    024D 600      ;
    50 10 A5 B2 024D 601      MCOMW BQO$W_VERSION(R5),R0 ;GET VMB VERSION NUMBER 1'S COMPLEMENTED
    12 A5 50 B1 0251 602      CMPW R0,BQO$W_VERSION+2(R5) ;CHECK AGAINST CHECK WORD IN VMB
    03 10 A5 B1 0255 603      BNEQ 40$ ;IF NOT, ASSUME NO VERSION NUMBER
    50 0D DB 0257 604      CMPW BQO$W_VERSION(R5),#3 ;VERSION 3 OF VMB?
    51 0C DB 025B 605      BGEQU 50$ ;IF OK, USE DESCRIPTORS IN RPB
    50 0D DB 025D 606 40$:      MFPR #PR$SLR,R0 ;GET LENGTH OF SPT IN LONGWORDS
    51 0C DB 0260 607      MFPR #PR$SBR,R1 ;GET PHYSICAL ADDRESS OF SPT
    00BC C6 50 F7 8F DE 0263 608      MOVAL (R1)[R0],R0 ;COMPUTE TOTAL PHYSICAL MEMORY SIZE
    026E 609      ASHL #-9,R0,RPB$SL_MEMDSC(R6) ;STORE IN MEM. DESCRIPTOR PAGCNT,TR=0
    026E 610      ASSUME RPB$V_PAGCNT EQ 0
    026E 611      ASSUME RPB$V_TR EQ <RPB$V_PAGCNT+24>
```

```
00C0 C6 7C 026E 612 ASSUME RPB$C_MEMDSC$SIZ EQ 8
026E 613 CLRQ RPB$L_MEMDSC+4(R6) ;SET STARTPFN=0 AND STORE 0 TERMINATOR
0272 614 ASSUME RPB$V_BASEPFN EQ 32
0272 615 ASSUME RPB$C_NMEMDSC GE 2
0272 616
0272 617 :: COPY THE MEMORY DESCRIPTORS FROM THE RPB INITIALIZED IN VMB
0272 618
0272 619
0272 620 ASSUME RPB$C_NMEMDSC EQ DMP$C_NMEMDSC
0272 621 ASSUME RPB$C_MEMDSC$SIZ EQ DMP$C_MEMDSC$SIZ
63 00BC C6 0040 8F 28 0272 621 50$: MOVCL #<RPB$C_NMEMDSC*RPB$C_MEMDSC$SIZ>,RPB$L_MEMDSC(R6),(R3) ;SET THE
027A 622 ;MEMORY DESCRIPTORS FROM THE RPB
027A 623
027A 624 :: STORE THE SYSTEM VERSION AND ONE'S COMPLEMENT CHECKSUM IN HEADER
027A 625
027A 626
027A 627 MOVCL #SYS$K_VERSION,(R3) ;SET THE VERSION # OF THE SYSTEM
0281 628 MCOML (R3),4(R3) ;SET CHECK FIELD = ONES COMPLEMENT
0285 629
0285 630 :: WRITE THE FIRST BLOCK OF THE DUMP FILE (THE HEADER)
0285 631
0285 632 MOVZWL #<EMBSK_CR_LENGTH+EMBSK_LENGTH+DMP$C_LENGTH>,R8 ;BUFFER SIZE
58 017C 8F 3C 028A 632 MOVCL BOO$L_DMP_MAP(R10),R5 ;VIRTUAL TO LOGICAL MAP FOR DUMP FILE
55 20 AA D0 028E 633 MOVCL BOO$L_DMP_VBN(R10),R10 ;STARTING VBN OF DUMP FILE
5A 18 AA D0 0292 634 BSBW WRITEDUMP ;WRITE DUMP HEADER
00E6 30 0295 635
0295 636 :: WRITE THE NEXT 2 BLOCKS OF ERROR LOG BUFFERS
0295 637
0295 638
0295 639 MOVZWL #<2*512>,R8 ;SET SIZE FOR ERROR LOG BUFFERS
58 0400 8F 3C 029A 639 MOVCL @#ERL$AL_BUFADDR,R7 ;AND BUFFER ADDRESS
57 00000000'9F D0 02A1 640 BSBW WRITEDUMP ;WRITE ERROR LOG BUFFERS
00D7 30 02A4 641
02A4 642 :: NOW WRITE EVERY PAGE OF EVERY MEMORY OUT TO THE DUMP FILE. VMB HAS BUILT
02A4 643 MEMORY DESCRIPTORS INTO THE RPB. EACH DESCRIPTOR GIVES THE TR NUMBER, BASE
02A4 644 PFN, AND PAGE COUNT FOR A PARTICULAR MEMORY. THERE MAY BE UP TO EIGHT MEMORY
02A4 645 DESCRIPTORS. A DESCRIPTOR WITH A ZERO PAGE COUNT AND TR NUMBER INDICATES NO
02A4 646 MORE DESCRIPTORS.
02A4 647
02A4 648
02A4 649 ASSUME RPB$C_MEMDSC$SIZ EQ 8
58 54 08 D0 02A4 649 MOVCL #RPB$C_NMEMDSC,R4 ;GET MAXIMUM # OF MEM DESC POSSIBLE
58 00BC C6 9E 02A7 650 MOVAB RPB$L_MEMDSC(R6),R11 ;GET ADR OF FIRST MEM DESC
68 18 00 EF 02AC 651 60$: EXTZV #RPB$V_PAGECNT,#RPB$S_PAGECNT,(R11),R8 ;GET PAGECNT FOR THIS MEM
58 58 09 78 02B1 652 BEQL NODUMP ;BR IF MEM DSC NOT USED
02B3 653 ASHL #9,R8,R8 ;CONVERT PAGE COUNT TO BYTE COUNT
02B7 654 ASSUME <RPB$S_PAGECNT + RPB$S_TR> EQ 32
58 04 C0 02B7 655 ADDL #4,R11 ;POINT TO BASE PFN IN MEMORY DESC
02BA 656 ASSUME RPB$S_BASEPFN EQ 32
02BA 657 ASSUME RPB$V_BASEPFN EQ 32
57 57 88 D0 02BA 658 MOVCL (R11),R7 ;GET BASE PFN FOR THIS MEMORY
57 57 09 78 02BD 659 ASHL #9,R7,R7 ;CONVERT PFN TO PHYSICAL ADDRESS
00B7 30 02C1 660 BSBW WRITEDUMP ;DUMP PAGES FOR THIS MEMORY
E5 54 F5 02C4 661 SOBGTR R4,60$ ;LOOK FOR ANOTHER MEMORY DESCRIPTOR
02C7 662
02C7 663 :: CHECK THE VMB VERSION NUMBER. IF IT EXISTS AND IF IT IS 10 OR GREATER,
02C7 664 THEN CALL A UNIT DISCONNECT ROUTINE TO DO ANY DEVICE/UNIT SPECIFIC
02C7 665 DISCONNECT/DISMOUNT FOR THE SYSTEM DEVICE.
02C7 666
02C7 667
02C7 668 NODUMP: MOVCL @#EXE$GL_RPB,R9 ;PICK UP THE RPB POINTER
59 00000000'9F D0 02C7 668
```

- SOFTWARE BUG CHECK ERROR LOGIC
NON-RESIDENT BUG CHECK CODE

```
16-SEP-1984 02:37:19 VAX/VMS Macro V04-00
5-SEP-1984 03:40:15 [SYS.SRC]BUGCHECK.MAR:1
```

Page 14
(1)

51	34	A9	D0	02CE	669		MOVL	RPS\$L_IOVEC(R9),R1	:FETCH POINTER TO BOOTDRIVER
50	10	A1	B2	02D2	670		MCMW	BQO\$W_VERSION(R1),R0	:GET VMB VERSION NUMBER 1'S COMPLEMENTED
12	A1	50	B1	02D6	671		CMPW	R0,BQO\$W_VERSION+2(R1)	:CHECK AGAINST CHECK WORD IN VMB
		10	12	02DA	672		BNEQ	10\$:IF NOT, ASSUME NO VERSION NUMBER
0A	10	A1	B1	02DC	673		CMPW	BQO\$W_VERSION(R1),#10	:VERSION 10 OR GREATER OF VMB?
		0A	1F	02E0	674		BLSSU	10\$:NO, DON'T CALL THE NON-EXISTENT CODE
55	2C	A1	D0	02E2	675		MOVL	BQO\$L_UNIT_DISC(R1),R5	:YES, IS THE ROUTINE PRESENT?
		04	13	02E6	676		BEQL	10\$:NO, DON'T CALL
6145		00	FB	02E8	677		CALLS	#0,(R1)[R5]	:DO IT!
				02EC	678	:			
				02EC	679	::			
				02EC	680	:			
06	00000000	'9F	00'	E1	02EC	681	10\$:	BBC	S^#EXESV_BUGREBOOT,@#EXESGL_FLAGS,20\$:BRANCH IF NO REBOOT
	0000015E	'9F	17	D4	02F4	682		JMP	@#REBOOT :REBOOT THE PROCESSOR
		5B	D4	02FA	683	20\$:		CLRL	R11 :SET FOR CONSOLE TERMINAL OUTPUT
51	0000004C	'EF	9E	02FC	684			MOVAB	SHUT_DOWN,R1 :SET ADDRESS OF MESSAGE STRING
		FCFA	30	0303	685			BSBW	EXESOUTZSTRING :AND OUTPUT IT TO THE CONSOLE
		FE	11	0306	686	30\$:		BRB	30\$:LOOP FOREVER
				0308	687				

```
0308 689 .SBTTL DUMP_ARRAY - SUBROUTINE TO DUMP AN ARRAY OF MEMORY LOCATIONS
0308 690 :
0308 691 : DUMP AN ARRAY OF MEMORY LOCATIONS WITH THEIR ADDRESSES AND CONTENTS
0308 692 :
0308 693 : INPUTS:
0308 694 :
0308 695 : R8 = NUMBER OF LONG WORDS TO DUMP
0308 696 : IF 0 IS SPECIFIED THE FIRST ADDRESS IS CHECKED FOR RESIDENCY
0308 697 : AP = ADDRESS OF FIRST LONG WORD TO DUMP
0308 698 :
0308 699 : OUTPUTS:
0308 700 :
0308 701 : R0 = LOW BIT SET IF SUCCESSFUL
0308 702 : = LOW BIT CLEAR IF CANNOT ACCESS THE ADDRESS IN AP
0308 703 : AP = ADDRESS OF NEXT LONG WORD NOT DUMPED
0308 704 : R4,R5,R8 ALTERED
0308 705 :
0308 706 DUMP_ARRAY:
54 00000000'9F 9E 0308 707 MOVAB @#MMG$AL SYSPCB,R4 ;PCB ADDRESS IF SYSTEM SPACE
07 5C 1F E0 030F 708 BBS #31,AP,20$ ;BRANCH IF SYSTEM SPACE
54 00000000'9F D0 0313 709 MOVL @#SCH$GL CURPCB,R4 ;PROCESS PCB FOR PROCESS SPAC
55 6C A4 D0 031A 710 20$: MOVL PCBSL_PHD(R4),R5 ;CORRESPONDING PROCESS HEADER ADDRESS
1B 11 031E 711 70$ ;LOOP 0 OR MORE TIMES
50 09 9A 0320 712 60$: MOVZBL #^A/ /,R0 ;GET TAB CHARACTER
FCDA' 30 0323 713 BSBW EXE$OUTCHAR ;OUTPUT TAB CHARACTER
51 5C D0 0326 714 MOVL AP,R1 ;GET ADDRESS OF LONGWORD TO CONVERT
FCD4' 30 0329 715 BSBW EXE$OUTHEX ;CONVERT ADDRESS OF LONGWORD
FCD1' 30 032C 716 BSBW EXE$OUTBLANK ;OUTPUT BLANK CHARACTER
FCCE' 30 032F 717 BSBW EXE$OUTBLANK ;OUTPUT BLANK CHARACTER
51 8C D0 0332 718 MOVL (AP)+,R1 ;GET CONTENTS OF LONGWORD TO OUTPUT
FCC8' 30 0335 719 BSBW EXE$OUTHEX ;OUTPUT CONVERTED HEX LONGWORD
FCC5' 30 0338 720 BSBW EXE$OUTCRLF ;OUTPUT CARRIAGE RETURN, LINE FEED PAIR
52 5C D0 033B 721 70$: MOVL AP,R2 ;MAKE SURE THAT THIS ADDRESS CAN BE ACCESSED
00000000'9F 16 033E 722 JSB @#MMG$PTEINDX ;GET LONG WORD INDEX TO SVAPTE IN R3
31 50 E9 0344 723 BLBC R0,100$ ;BRANCH IF LENGTH VIOLATION
53 6543 DE 0347 724 MOVAL (R5)[R3],R3 ;FORM SYSTEM VIRTUAL ADR OF PTE
63 D5 034B 725 TSTL (R3) ;SEE IF PAGE TABLE ENTRY IS VALID
22 19 034D 726 BLSS 75$ ;BRANCH IF IT IS
02 A3 0440 8F B3 034F 727 BITW #<PTESM_TYP1 ! PTESM_TYPO>@-16,2(R3) ;IF TRANSITION PAGE
21 12 0355 728 BNEQ 100$ ;BRANCH IF NOT
50 63 15 00 EF 0357 729 EXTZV #PTESV_PFN,#PTESS_PFN,(R3),R0 ;GET PAGE FRAME NUMBER
1A 13 035C 730 BEQL 100$ ;BRANCH IF DEMAND ZERO FORMAT
00000000'9F DD 035E 731 PUSHL @#PFNS$AB STATE ;ADDRESS OF STATE TABLE
03 00 ED 0364 732 CMPZV #PFNS$V_LOC,#PFNS$S_LOC,- ;PAGE IS OK IN MEMORY UNLESS
06 9E40 0367 733 @ (SP)+[R0],#PFNS$C_RDINPROG ;IT IS BEING READ IN
03 A3 80 8F 88 036A 734 BEQL 100$
AC 58 F4 0371 735 BISB #<PTESM_VALID>@-24,3(R3) ;JAM IT VALID AND USE IT
50 01 D0 0374 736 75$: SOBGEQ R8,60$ ;ANY MORE LONGWORDS TO CONVERT?
05 05 0377 737 80$: MOVL #1,R0 ;INDICATE SUCCESSFUL COMPLETION
0378 738 RSB
0378 739 :
0378 740 : CANNOT ACCESS ADDRESS POINTED TO BY AP
0378 741 :
0378 742 100$: CLRL R0
05 037A 743 RSB
```

```
037B 745 .SBTTL WRITEDUMP - WRITE DATA TO DUMP FILE
037B 746
037B 747 WRITE DATA TO SYSTEM DUMP FILE
037B 748
037B 749 INPUTS:
037B 750 R5 - ADDRESS OF VIRTUAL TO LOGICAL MAP FOR DUMP FILE
037B 751 R6 - ADDRESS OF RESTART PARAMETER BLOCK
037B 752 R7 - BUFFER ADDRESS
037B 753 R8 - SIZE OF BUFFER IN BYTES
037B 754 R9 - NUMBER OF BLOCKS NOT YET WRITTEN IN DUMP FILE
037B 755 R10 - VBN OF DUMP FILE (UPDATED)
037B 756
037B 757 OUTPUTS:
037B 758 R7 - UPDATED
037B 759 R8 - UPDATED
037B 760 R9 - UPDATED
037B 761 R10 - UPDATED
037B 762
0000FE00 037B 763 IOSIZE=127*512 ;MAXIMUM TRANSFER SIZE
037B 764 WRITEDUMP: ;SAVE MAP AND VBN
037B 765 PUSH R5,R10 ;COUNT OF RETRIEVAL POINTERS
037B 766 ASHL #3,(R5)+,R2 ;R0=BLOCK COUNT, R1=STARTING LBN
037B 767 10$: MOVQ (R5)+,R0 ;VBN COVERED BY THIS RETRIEVAL POINTER?
037B 768 CMPL R10,R0 ;BRANCH IF YES
037B 769 BLEQ 20$ ;NO, REDUCE VBN BY BLOCKS PASSED OVER
037B 770 SUBL R0,R10 ;TRY NEXT RETRIEVAL POINTER
037B 771 SOBGTR R2,10$ ;RESTORE MAP AND VBN
037B 772 POP R5,R10 ;EOF, NO MORE WRITING
037B 773 BRB 100$ ;MAKE VBN 0 ORIGIN
037B 774 20$: DECL R10 ;NO. OF BLOCKS AFTER DESIRED VBN
037B 775 SUBL R10,R0 ;STARTING LBN OF DESIRED VBN
037B 776 ADDL R10,R1 ;RESTORE MAP AND VBN
037B 777 POP R5,R10
037B 778
037B 779 R0 = NUMBER OF BLOCKS THAT COULD BE TRANSFERRED
037B 780 R1 = STARTING LBN OF THE TRANSFER
037B 781
037B 782 MOVZWL #IOSIZE,R3 ;ASSUME MAXIMUM
037B 783 ASHL #9,R0,R0 ;BYTE COUNT THAT COULD BE TRANSFERRED
037B 784 CMPL R3,R0 ;MINIMIZE WITH MAX LEGAL XFER
037B 785 BLEQ 30$
037B 786 MOVL R0,R3
037B 787 30$: CMPL R3,R8 ;MINIMIZE WITH BYTE COUNT
037B 788 BLEQ 40$ ;REMAINING TO BE TRANSFERRED
037B 789 MOVL R8,R3
037B 790 40$: MOVAB 511(R3),R2 ;ROUND UP BYTE COUNT AND FORM
037B 791 ASHL #-9,R2,R2 ;PAGES TO BE WRITTEN
037B 792 BEQL 100$ ;NOTE NOTHING TO TRANSFER
037B 793 CMPL R2,R9 ;MINIMIZE WITH PAGES LEFT IN FILE
037B 794 BLEQ 50$
037B 795 ASHL #9,R9,R3 ;USE BYTE COUNT REMAINING IN FILE
037B 796 MOVL R9,R2 ;AND BLOCK COUNT TO TRANSFER
037B 797 BEQL 100$ ;BRANCH IF NO BLOCK LEFT IN FILE
037B 798 50$: PUSH R6 ;SET ADDRESS OF RPB
037B 799 EXTZV #VASV SYSTEM,#1,R7,-(SP) ;USE SYSTEM BIT AS VIRTUAL FLAG
037B 800 PUSH R1 ;SET FUNCTION CODE
037B 801 PUSH R1 ;LBN IN DUMP FILE
```

52 85 0420 8F BB 037B 765
FD 8F 78 037F 766
50 85 7D 0384 767 10\$:
50 5A D1 0387 768
OC 15 038A 769
5A 50 C2 038C 770
F2 52 F5 038F 771
0420 8F BA 0392 772
6A 11 0396 773
5A D7 0398 774 20\$:
50 5A C2 039A 775
51 5A C0 039D 776
0420 8F BA 03A0 777
03A4 778
03A4 779
03A4 780
03A4 781
53 FE00 8F 3C 03A4 782
50 50 09 78 03A9 783
50 53 D1 03AD 784
03 15 03B0 785
53 50 D0 03B2 786
58 53 D1 03B5 787 30\$:
03 15 03B8 788
53 58 D0 03BA 789
52 01FF C3 9E 03BD 790 40\$:
52 F7 8F 78 03C2 791
39 13 03C7 792
59 52 D1 03C9 793
09 15 03CC 794
53 59 09 78 03CE 795
52 59 D0 03D2 796
2B 13 03D5 797
56 DD 03D7 798 50\$:
7E 57 01 1F EF 03D9 799
20 DD 03DE 800
51 DD 03E0 801

50	34	A6	DD	03E2	802	PUSHL	R3	:SIZE OF BUFFER IN BYTES
00	8040	06	DD	03E4	803	PUSHL	R7	:ADDRESS OF BUFFER
57	53	06	DO	03E6	804	MOVL	RPB\$L_IOVEC(R6),R0	:BOOT DRIVER VECTOR
5A	53	06	FB	03EA	805	CALLS	#6,2800SL_QIO(R0)[R0]	:CALL BOOTDRVR
59	52	06	CO	03EF	806	ADDL	R3,R7	:UPDATE BUFFER ADDRESS
	52	06	CO	03F2	807	ADDL	R2,R10	:UPDATE VBN
	52	06	C2	03F3	808	SUBL	R2,R9	:AND SIZE OF FILE
	08	06	15	03F8	809	BLEQ	100\$:DONE IF END OF FILE
58	03	06	C2	03FA	810	SUBL	R3,R8	:UPDATE BYTE COUNT
	03	06	15	03FD	811	BLEQ	100\$:DONE IF BYTE COUNT EXHAUSTED
	FF79	06	31	03FF	812	BRW	WRITEDUMP	:OTHERWISE START FROM THE TOP
		06	05	0402	813	RSB		:

100\$:

```
0403 815 .SBTTL SUBROUTINES TO BUILD HEADERS AND VERIFY BOOT CONTROL BLOCK
0403 816
0403 817
0403 818 : SUBROUTINE TO BUILD HEADER AND DUMP GENERAL REGISTERS
0403 819 :
0403 820 :
0000 0178 821 .PSECT $AEXENONPAGED
0178 822 BUILD_HEADER:
0178 823 ASSUME EMB$W_BC_ENTRY EQ EMB$W_CR_ENTRY
04 A2 59 B0 0178 824 MOVW R9, EMB$W_BC_ENTRY(R2) ; SET TYPE OF ENTRY IN EMB
017C 825 ASSUME EMB$L_BC_KSP EQ EMB$L_CR_KSP
50 10 A2 9E 017C 826 MOVAB EMB$L_BC_KSP(R2), R0 ; POINT TO PLACE IN EMB FOR 1ST REGISTER
0A 10 0180 827 BSBB DUMP_REGISTERS ; INSERT PROCESSOR STACK POINTERS
0182 828 ASSUME EMB$C_BC_R0 EQ EMB$L_BC_R0
0182 829 ASSUME EMB$L_BC_PSL EQ EMB$C_CR_PSL
51 11 D0 0182 830 MOVL #<EMB$L_BC_PSL+4-EMB$C_BC_R0>/4, R1 ; SET NUMBER OF REGISTERS
80 8C D0 0185 831 10$: MOVL (AP)+, (R0) ; INSERT GENERAL REGISTER
FA 51 F5 0188 832 SOBGTR R1, 10$ ; ANY MORE REGISTERS TO INSERT?
05 018B 833 RSB
018C 834
018C 835 : SUBROUTINE TO DUMP PROCESSOR REGISTERS UNTIL ESCAPE
018C 836 :
018C 837 :
018C 838 :
018C 839 DUMP_REGISTERS:
51 8B 98 018C 840 CVTBL (R11)+, R1 ; GET NEXT INTERNAL REGISTER NUMBER
05 19 018F 841 BLSS RETURN ; IF LSS ESCAPE
80 51 DB 0191 842 MFPR R1, (R0)+ ; INSERT PROCESSOR REGISTER
F6 11 0194 843 BRB DUMP_REGISTERS
05 0196 844 RETURN: RSB
0197 845 :
0197 846 : VALIDATE THE CHECKSUM FOR THE BOOT CONTROL BLOCK
0197 847 : AND SYS.EXE WINDOW CONTROL BLOCK
0197 848 :
0197 849 INPUTS:
0197 850 :
0197 851 EXE$GQ_BOOTCB_D a descriptor for the boot control block and SYS.EXE
0197 852 : window control block. The descriptor is assumed to
0197 853 : delineate an area which includes both the boot control
0197 854 : block and the SYS.EXE window control block. The
0197 855 : address field of the descriptor is assumed to point to
0197 856 : the boot control block.
0197 857 :
0197 858 IPL >= IPL$_SYNCH (If IPL is lower than IPL$_SYNCH, the checksum
0197 859 : calculation may be wrong.)
0197 860 :
0197 861 OUTPUTS:
0197 862 :
0197 863 Z SET IF CHECKSUM MATCHES, Z CLEAR IF NOT
0197 864 R1 = ADDRESS OF BEGINNING OF BOOT CONTROL BLOCK
0197 865 R3 = DESIRED CHECKSUM VALUE
0197 866 R0 ALTERED
0197 867 ALL OTHER REGISTERS PRESERVED
0197 868 :
0197 869 :
0197 870 EXE$BOOTCB_CHK::
50 00000000'EF 7D 0197 871 MOVQ EXE$GQ_BOOTCB_D, R0 ; GET DESCRIPTOR OF BLOCK TO CHECKSUM
```

```
51 50 C0 019E 872 ADDL R0,R1 ;POINT TO END OF BOOT CONTROL BLOCK
50 04 C6 01A1 873 DIVL #4,R0 ;FORM LONG WORD COUNT
      01A4 874
      01A4 875
      50 D7 01A4 876
      53 D4 01A6 877
53 71 C0 01A8 878 10$:
      FA 50 F5 01AB 879
50 10 A1 D0 01AE 880
53 24 A0 C2 01B2 881
53 28 A0 C2 01B6 882
71 53 D1 01BA 883
      05 01BD 884
      01BE 885
      01BE 886
      .END
```

```
ASSUME BOO$$_CHECKSUM EQ 0
DECL R0
CLRL R3
ADDL -(R1),R3
SOBGT R0,10$
MOVL BOO$$_SYS_MAP-4(R1),R0
SUBL WCB$$_READS(R0),R3
SUBL WCB$$_WRITES(R0),R3
CMPL R3,-(R1)
RSB
```

```
;DON'T ADD FIRST LONG WORD
;INIT CHECKSUM
;FORM ADDITIVE CHECKSUM
;LOOP THROUGH THE BLOCK
; Get pointer to system WCB.
; Remove to varying WCB entries from
; the checksum.
;DOES THE CHECKSUM MATCH
```

BUGCHECK
Symbol table

- SOFTWARE BUG CHECK ERROR LOGIC

K 4

16-SEP-1984 02:37:19 VAX/VMS Macro V04-00
5-SEP-1984 03:40:15 [SYS.SRC]BUGCHECK.MAR;1Page 20
(1)

BOOSL_BUG_MAP	=	00000024		
BOOSL_CHECKSUM	=	00000000		
BOOSL_DMP_MAP	=	00000020		
BOOSL_DMP_SIZE	=	0000001C		
BOOSL_DMP_VBN	=	00000018		
BOOSL_SYS_MAP	=	00000014		
BQOSL_QIO	=	00000000		
BQOSL_UNIT_DISC	=	0000002C		
BQOSL_UNIT_INIT	=	0000001C		
BQOSW_VERSION	=	00000010		
BUGSA_PAGED	00000000	RG	07	
BUGSA_PAGEDEND	00000000	RG	04	
BUGSFATAL	00000000	RG	03	
BUGST_MESSAGES	*****	X	07	
BUGS_OPERATOR	*****	X	07	
BUGCHK_FLAGS	00000000	R	02	
BUILD_READER	00000178	R	06	
CONSC_BOOTCPU	=	00000002		
CONSOQNTY	*****	X	07	
CONSSSENDCONSCMD	*****	X	06	
CONSOLE_DONE	000001E7	R	07	
CR	=	0000000D		
CTLSAL_STACK	*****	X	07	
CTLSAL_STACKLIM	*****	X	07	
CTLSGL_IMGHDRBF	*****	X	07	
DMPSC_LENGTH	=	0000006C		
DMPSC_MEMDSCSIZ	=	00000008		
DMPSC_NMEMDSC	=	00000008		
DMPSL_ERRSEQ	=	00000000		
DMPSL_ESP	=	00000014		
DMPSL_ISP	=	00000020		
DMPSL_KSP	=	00000010		
DMPSL_SBR	=	00000008		
DMPSL_SLR	=	0000000C		
DMPSL_SSP	=	00000018		
DMPSL_USP	=	0000001C		
DMPSW_DUMPVER	=	00000006		
DMPSW_FLAGS	=	00000004		
DUMP_ARRAY	00000308	R	07	
DUMP_REGISTERS	0000018C	R	06	
EMBSB_VALID	=	FFFFFFFF		
EMBSK_BC_LENGTH	=	00000080		
EMBSK_CR	=	00000025		
EMBSK_CR_LENGTH	=	0000010C		
EMBSK_LENGTH	=	00000004		
EMBSK_SBC	=	00000028		
EMBSK_UBC	=	00000070		
EMBSL_BC_CODE	=	00000068		
EMBSL_BC_KSP	=	00000010		
EMBSL_BC_PID	=	0000006C		
EMBSL_BC_PSL	=	00000064		
EMBSL_BC_RO	=	00000024		
EMBSL_CR_CODE	=	000000F4		
EMBSL_CR_KSP	=	00000010		
EMBSL_CR_PID	=	000000F8		
EMBSL_CR_PSL	=	00000064		
EMBSL_HD_SID	=	00000000		

EMBSQ_CR_TIME	=	00000006		
EMBST_BC_LNAME	=	00000070		
EMBST_CR_LNAME	=	000000FC		
EMBSW_BC_ENTRY	=	00000004		
EMBSW_CR_ENTRY	=	00000004		
EMBSW_CR_ERRSEQ	=	0000000E		
EMBSW_SIZE	=	FFFFFFFF		
ERLSALOCMB	*****	X	06	
ERLSAL_BUFADDR	*****	X	07	
ERLSGL_SEQUENCE	*****	X	07	
ERLSRELEASEMB	*****	X	06	
EXESACVIOLAT	*****	X	06	
EXESAL_STACKS	*****	X	07	
EXESBOOTCB_CHK	00000197	RG	06	
EXESBUG_CHECK	00000012	RG	06	
EXESDUMPCPUREG	*****	X	07	
EXESGL_BOOTCB	*****	X	06	
EXESGL_BUGCHECK	00000008	RG	02	
EXESGL_FLAGS	*****	X	06	
EXESGL_RPB	*****	X	06	
EXESGQ_BOOTCB_D	*****	X	06	
EXESGQ_SYSTIME	*****	X	07	
EXESINIBOOTADP	*****	X	06	
EXESOUTBLANK	*****	X	07	
EXESOUTCHAR	*****	X	07	
EXESOUTCRLF	*****	X	07	
EXESOUTCSTRING	*****	X	07	
EXESOUTHEX	*****	X	07	
EXESOUTZSTRING	*****	X	07	
EXESSHUTDOWNADP	*****	X	06	
EXESV_BUGDUMP	*****	X	07	
EXESV_BUGREBOOT	*****	X	06	
EXESV_FATAL_BUG	*****	X	06	
EXESV_SYSWRTABL	*****	X	06	
FATALBUG	00000000	R	07	
FATAL_SPSAV	00000004	R	02	
IFDSW_FILNAMOFF	=	00000002		
IMGNAM_MSG	00000037	R	05	
INISBRK	*****	X	06	
INISWRITABLE	*****	X	06	
IOS_READLBLK	=	00000021		
IOS_WRITEBLK	=	00000020		
IOSIZE	=	0000FE00		
LF	=	0000000A		
MMGSAL_SYSPCB	*****	X	07	
MMGSPTINDEX	*****	X	07	
MPHSBUGCHKHK	000000F6	RG	06	
MSGCTRL	00000087	R	05	
MSGCTRL1	000000AB	R	05	
NODUMP	000002C7	R	07	
PCBSL_PHD	=	0000006C		
PCBSL_PID	=	00000060		
PCBSQ_PRIV	=	00000084		
PCBST_LNAME	=	00000070		
PFNSAB_STATE	*****	X	07	
PFNSC_RDINPROG	=	00000006		
PFNSS_LOC	=	00000003		

BUGCHECK
Symbol table

- SOFTWARE BUG CHECK ERROR LOGIC

L 4

16-SEP-1984 02:37:19 VAX/VMS Macro V04-00
5-SEP-1984 03:40:15 [SYS.SRC]BUGCHECK.MAR;1Page 21
(1)

PFNSV_LOC	= 00000000		
PRS_ASTLVL	= 00000013		
PRS_ESP	= 00000001		
PRS_ICCS	= 00000018		
PRS_IPL	= 00000012		
PRS_ISP	= 00000004		
PRS_KSP	= 00000000		
PRS_POBR	= 00000008		
PRS_POLR	= 00000009		
PRS_P1BR	= 0000000A		
PRS_P1LR	= 0000000B		
PRS_PCBB	= 00000010		
PRS_SBR	= 0000000C		
PRS_SCBB	= 00000011		
PRS_SID	= 0000003E		
PRS_SISR	= 00000015		
PRS_SLR	= 0000000D		
PRS_SSP	= 00000002		
PRS_USP	= 00000003		
PRCNAM_MSG	00000000	R	05
PRCPRV_MSG	0000001A	R	05
PRVSV_BUGCHK	= 00000017		
PSLSC_EXEC	= 00000001		
PSLSS_PRVMOD	= 00000002		
PSLSV_PRVMOD	= 00000016		
PTESM_TYPO	= 00400000		
PTESM_TYPI	= 04000000		
PTESM_VALID	= 80000000		
PTESS_PFN	= 00000015		
PTESV_PFN	= 00000000		
READ_ERR_RETRY	00000158	R	06
REBOOT	0000015E	R R	06
REGTAB	00000000	R R	06
RETURN	00000196	R	06
RPBSC_MEMDSCSI2	= 00000008		
RPBSC_NMEMDSC	= 00000008		
RPBSL_IOVEC	= 00000034		
RPBSL_MEMDSC	= 000000BC		
RPBSS_BASEPFN	= 00000020		
RPBSS_PAGCNT	= 00000018		
RPBSS_TR	= 00000008		
RPBSV_BASEPFN	= 00000020		
RPBSV_PAGCNT	= 00000000		
RPBSV_TR	= 00000018		
SCHSGC_CURPCB	*****	X	06
SCSSSHUTDOWN	*****	X	06
SHUT_DOWN	0000004C	R	05
SS\$ BUGCHECK	= 000002A4		
STSSK_ERROR	= 00000002		
STSSK_SEVERE	= 00000004		
STSSS_SEVERITY	= 00000003		
STSSV_SEVERITY	= 00000000		
SYSEXIT	*****	GX	06
SYSGQ_VERSION	*****	X	07
SYSSK_VERSION	*****	X	07
VASV_SYSTEM	= 0000001F		
WCB\$C_READS	= 00000024		

WCB\$C WRITES
WRITEDUMP= 00000028
0000037B R 07

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$025	00000000 (12.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$ZBUGFATAL	00000000 (0.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC WORD
Z\$INIT__BUGZEND	00000000 (0.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC WORD
Z\$INIT__BUGC	00000169 (361.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$AEXENORPAGED	000001BE (446.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
Z\$INIT__BUGA	00000403 (1027.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.08	00:00:00.25
Command processing	105	00:00:00.55	00:00:01.09
Pass 1	495	00:00:19.22	00:00:22.42
Symbol table sort	0	00:00:03.09	00:00:03.31
Pass 2	183	00:00:03.93	00:00:04.37
Symbol table output	21	00:00:00.18	00:00:00.18
Psect synopsis output	3	00:00:00.05	00:00:00.05
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	838	00:00:27.10	00:00:31.68

The working set limit was 1950 pages.

110864 bytes (217 pages) of virtual memory were used to buffer the intermediate code.

There were 110 pages of symbol table space allocated to hold 1960 non-local and 56 local symbols.

886 source lines were read in Pass 1, producing 29 object records in Pass 2.

37 pages of virtual memory were used to define 36 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	21
_\$255\$DUA28:[SYS.LIB]STARLET.MLB;2	12
TOTALS (all libraries)	33

2084 GETS were required to define 33 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:BUGCHECK/OBJ=OBJ\$:BUGCHECK MSRC\$:BUGCHECK/UPDATE=(ENH\$:BUGCHECK)+EXECML\$/LIB

0373

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY